

## Задача А. Снеговика

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	4 секунды
Ограничение по памяти:	64 мегабайта

Зима. 2012 год. На фоне грядущего Апокалипсиса и конца света незамеченной прошла новость об очередном прорыве в областях клонирования и снеговиков: клонирования снеговиков. Вы конечно знаете, но мы вам напомним, что снеговик состоит из нуля или более вертикально поставленных друг на друга шаров, а клонирование — это процесс создания идентичной копии (клона).

В местечке Местячково учитель Андрей Сергеевич Учитель купил через интернет-магазин «Интернет-магазин аппаратов клонирования» аппарат для клонирования снеговиков. Теперь дети могут играть и даже играют во дворе в следующую игру. Время от времени один из них выбирает понравившегося снеговика, клонирует его и:

- либо добавляет ему сверху один шар;
- либо удаляет из него верхний шар (если снеговик не пустой).

Учитель Андрей Сергеевич Учитель записал последовательность действий и теперь хочет узнать суммарную массу всех построенных снеговиков.

### Формат входных данных

Первая строка содержит количество действий  $n$  ( $1 \leq n \leq 200\,000$ ). В строке номер  $i + 1$  содержится описание действия  $i$ :

- $t\ m$  — клонировать снеговика номер  $t$  ( $0 \leq t < i$ ) и добавить сверху шар массой  $m$  ( $0 < m \leq 1000$ );
- $t\ 0$  — клонировать снеговика номер  $t$  ( $0 \leq t < i$ ) и удалить верхний шар. Гарантируется, что снеговик  $t$  не пустой.

В результате действия  $i$ , описанного в строке  $i + 1$  создается снеговик номер  $i$ . Изначально имеется пустой снеговик с номером ноль.

Все числа во входном файле целые.

### Формат выходных данных

Выведите суммарную массу построенных снеговиков.

### Пример

стандартный ввод	стандартный вывод
8	74
0 1	
1 5	
2 4	
3 2	
4 3	
5 0	
6 6	
1 0	

## Задача В. Персистентная очередь

Имя входного файла: `pqueue.in`  
Имя выходного файла: `pqueue.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 мегабайт

Персистентные структуры данных поддерживают доступ и модификацию любой версии структуры данных. В этой задаче вам нужно реализовать персистентную очередь.

Очередь — это структура данных, которая хранит список целых чисел и поддерживает две операции: `push` и `pop`. Операция `push(x)` добавляет  $x$  в конец списка. Операция `pop` возвращает первый элемент списка и удаляет его.

В персистентной версии очереди каждая операция принимает дополнительный аргумент  $v$ . Изначально очередь имеет версию 0. Рассмотрим  $i$ -ю операцию над очередью. Если это `push(v, x)`, то число  $x$  добавляется в конец  $v$ -й версии очереди и в полученная очередь становится версии  $i$  ( $v$ -я версия остается неизменной). Если это `pop(v)`, то первое число удаляется из  $v$ -й версии очереди и полученная очередь становится версии  $i$  (так же, версия  $v$  остается неизменной).

Задана последовательность операций над персистентной очередью, выведите результаты всех операций `pop`.

### Формат входных данных

Первая строка содержит целое число  $n$  — число операций ( $1 \leq n \leq 200\,000$ ). Следующие  $n$  строк описывают операции.  $i$ -я из этих строк задает  $i$ -ю операцию. Операция `push(v, x)` задается, как “ $1\ v\ x$ ”, операция `pop(v)` задается, как “ $-1\ v$ ”. Гарантировано, что операция `pop` никогда не применяется к пустой очереди. Элементы, добавляемые в очередь, влезают в стандартный знаковый 32-битный целочисленный тип данных.

### Формат выходных данных

Для каждой операции `pop` выведите значение элемента, который был извлечен из очереди.

### Примеры

<code>pqueue.in</code>	<code>pqueue.out</code>
10	1
1 0 1	2
1 1 2	3
1 2 3	1
1 2 4	2
-1 3	4
-1 5	
-1 6	
-1 4	
-1 8	
-1 9	

## Задача С. Двумерные запросы

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 мегабайт

Вам задан массив размера  $2^{17}$ . Требуется ответить на запросы: сколько есть элементов  $f[i]$  таких, что  $l \leq i \leq r$  и  $x \leq f[i] \leq y$ .

### Формат входных данных

На первой строке число  $q$  ( $1 \leq q \leq 2^{17}$ ). На второй строке пара целых чисел  $a, b$  от 1 до  $10^9$ , используемая в генераторе случайных чисел.

```
0. unsigned int a, b; // даны во входных данных
1. unsigned int cur = 0; // беззнаковое 32-битное число
2. unsigned int nextRand17() {
3.     cur = cur * a + b; // вычисляется с переполнениями
4.     return cur >> 15; // число от 0 до  $2^{17} - 1$ .
5. }
6. unsigned int nextRand24() {
7.     cur = cur * a + b; // вычисляется с переполнениями
8.     return cur >> 8; // число от 0 до  $2^{24} - 1$ .
9. }
```

Сначала массив генерируется следующим образом:

```
1. for (int i = 0; i < 1 << 17; i++)
2.     f[i] = nextRand24();
```

Потом генерируются запросы следующим образом:

```
1. l = nextRand17();
2. r = nextRand17();
3. if (l > r) swap(l, r); // получили отрезок [l..r]
4. x = nextRand24();
5. y = nextRand24();
6. if (x > y) swap(x, y); // получили отрезок [x..y]
7. b += c; // c -- ответ на данный запрос, для ответа на запросы в online
```

### Формат выходных данных

Выведите сумму ответов на все запросы второго типа по модулю  $2^{32}$ .

### Примеры

стандартный ввод	стандартный вывод
5 13 239	111139

## Задача D. $K$ -я порядковая статистика на отрезке

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 5 секунд  
Ограничение по памяти: 512 мегабайт

Дан массив из  $N$  неотрицательных чисел, строго меньших  $10^9$ . Вам необходимо ответить на несколько запросов о величине  $k$ -й порядковой статистики на отрезке  $[l, r]$ .

### Формат входных данных

Первая строка содержит число  $N$  ( $1 \leq N \leq 450\,000$ ) — размер массива.

Вторая строка может быть использована для генерации  $a_i$  — начальных значений элементов массива. Она содержит три числа  $a_1, l$  и  $m$  ( $0 \leq a_1, l, m < 10^9$ ); для  $i$  от 2 до  $N$

$$a_i = (a_{i-1} \cdot l + m) \bmod 10^9.$$

В частности,  $0 \leq a_i < 10^9$ .

Третья строка содержит одно целое число  $B$  ( $1 \leq B \leq 1000$ ) — количество групп запросов.

Следующие  $B$  строк описывают одну группу запросов. Каждая группа запросов описывается 10 числами. Первое число  $G$  обозначает количество запросов в группе. Далее следуют числа  $x_1, l_x$  и  $m_x$ , затем  $y_1, l_y$  и  $m_y$ , затем,  $k_1, l_k$  и  $m_k$  ( $1 \leq x_1 \leq y_1 \leq N$ ,  $1 \leq k_1 \leq y_1 - x_1 + 1$ ,  $0 \leq l_x, m_x, l_y, m_y, l_k, m_k < 10^9$ ). Эти числа используются для генерации вспомогательных последовательностей  $x_g$  и  $y_g$ , а также параметров запросов  $i_g, j_g$  и  $k_g$  ( $1 \leq g \leq G$ )

$$\begin{aligned}x_g &= ((i_{g-1} - 1) \cdot l_x + m_x) \bmod N + 1, & 2 \leq g \leq G \\y_g &= ((j_{g-1} - 1) \cdot l_y + m_y) \bmod N + 1, & 2 \leq g \leq G \\i_g &= \min(x_g, y_g), & 1 \leq g \leq G \\j_g &= \max(x_g, y_g), & 1 \leq g \leq G \\k_g &= (((k_{g-1} - 1) \cdot l_k + m_k) \bmod (j_g - i_g + 1)) + 1, & 2 \leq g \leq G\end{aligned}$$

Сгенерированные последовательности описывают запросы,  $g$ -й запрос состоит в поиске  $k_g$ -го по величине числа среди элементов отрезка  $[i_g, j_g]$ .

Суммарное количество запросов не превосходит 600 000.

### Формат выходных данных

Выведите единственное число — сумму ответов на запросы.

### Пример

стандартный ввод	стандартный вывод
5	15
1 1 1	
5	
1	
1 0 0 3 0 0 2 0 0	
1	
2 0 0 5 0 0 3 0 0	
1	
1 0 0 5 0 0 5 0 0	
1	
3 0 0 3 0 0 1 0 0	
1	
1 0 0 4 0 0 1 0 0	

## Задача E. Intercity Express

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Андрей разрабатывает систему для продажи железнодорожных билетов. Он собирается протестировать ее на Междугородней Экспресс линии, которая соединяет два больших города и имеет  $n - 2$  промежуточных станций, то есть в итоге есть  $n$  станций, пронумерованных от 1 до  $n$ .

В Междугороднем Экспресс поезде есть  $s$  мест, пронумерованных с 1 до  $s$ . В тестирующем режиме система имеет доступ к базе данных, содержащей проданные билеты в направлении от станции 1 до станции  $n$  и должна отвечать на вопросы, можно ли продать билет от станции  $a$  до станции  $b$ , и если да, нужно найти минимальный номер места, которое свободно на протяжении всего пути между  $a$  и  $b$ .

Изначально система имеет только доступ на чтение, то есть даже если есть свободное место, она должна сообщить об этом, но не должна изменять данные.

Помогите Андрею протестировать его систему написанием программы, которые будет находить ответы на вопросы.

### Формат входных данных

Первая строка содержит число  $n$  — количество станций,  $s$  — количество мест и  $m$  — количество уже проданных билетов ( $2 \leq n \leq 10^9$ ,  $1 \leq s \leq 100\,000$ ,  $0 \leq m \leq 100\,000$ ).

В следующих  $m$  строках описаны билеты, описание каждого билета состоит из трех чисел:  $c_i$ ,  $a_i$  и  $b_i$  — номер места, которое занимает владелец билета, номер станции, с которой продан билет и номер станции, до которой продан билет ( $1 \leq c_i \leq s$ ,  $1 \leq a_i < b_i \leq n$ ).

Следующая строка содержит число  $q$  — количество запросов ( $1 \leq q \leq 100\,000$ ). Специальное значение  $p$  должно поддерживаться в течение считывания запросов. Изначально  $p = 0$ .

Следующие  $2q$  целых чисел описывают запросы. Каждый запрос описывается двумя числами:  $x_i$  и  $y_i$  ( $x_i < y_i$ ).

Чтобы получить города  $a$  и  $b$  между которыми нужно проверить наличие места, используется следующая формула:

$a = x_i + p$ ,  $b = y_i + p$ . Ответ на запрос — число 0, если нет места на каждом отрезке между  $a$  и  $b$ , или минимальный номер свободного места.

После ответа на запрос, надо приравнять число  $p$  полученному ответу на запрос.

### Формат выходных данных

Для каждого запроса выведите ответ на него.

## Пример

стандартный ввод	стандартный вывод
5 3 5	1
1 2 5	2
2 1 2	2
2 4 5	3
3 2 3	0
3 3 4	2
10	0
1 2	0
1 2	0
1 2	0
2 3	
-2 0	
2 4	
1 3	
1 4	
2 5	
1 5	

## Замечание

Обратите внимание, что запросы выглядят так: (1, 2), (2, 3), (3, 4), (4, 5), (1, 3), (2, 4), (3, 5), (1, 4), (2, 5), (1, 5).

## Задача F. Запросы на треугольниках

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Ваша задача — написать программу, хранящую мультимножество точек и позволяющую отвечать на запросы двух видов:

- добавить точку в мультимножество,
- посчитать количество точек множества, лежащих внутри или на границе данного треугольника.

### Формат входных данных

В первой строке дано число запросов  $m$  ( $1 \leq m \leq 100000$ ). Следующие  $m$  строк содержат или  $1 \ x \ y$  или  $2 \ x \ y \ r$ . Запрос 2-го типа представлен треугольником с углами в точках  $(x, y)$ ,  $(x + r, y)$ ,  $(x, y + r)$ . Известно, что  $|x|, |y|, r \leq 10^8$ ,  $r > 0$

### Формат выходных данных

Для каждого запроса-треугольника в отдельной строке одно целое число – ответ на запрос.

### Пример

стандартный ввод	стандартный вывод
5	1
1 2 2	2
1 4 4	
1 6 6	
2 1 1 2	
2 1 1 6	

## Задача G. Соединение и разъединение

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Вы когда-нибудь слышали про обход в глубину? Например, используя этот алгоритм, вы можете проверить является ли граф связным за время  $O(E)$ . Вы можете даже посчитать количество компонент связности за то же время.

А вы когда-нибудь слышали про систему непересекающихся множеств? Используя эту структуру, вы можете быстро обрабатывать запросы “Добавить ребро в граф” и “Посчитать количество компонент связности в графе”.

А вы когда-нибудь слышали о *динамической* задаче связности? В этой задаче вам необходимо обрабатывать три типа запросов:

1. Добавить ребро в граф.
2. Удалить ребро из графа.
3. Посчитать количество компонент связности в графе.

Можно считать, что граф является неориентированным. Изначально граф является пустым.

### Формат входных данных

В первой строке находятся два целых числа  $N$  и  $K$  — количество вершин и количество запросов, соответственно ( $1 \leq N \leq 300\,000$ ,  $0 \leq K \leq 300\,000$ ). Следующие  $K$  строк содержат запросы, по одному в строке. Каждый запрос имеет один из трех типов:

1.  $+ u v$ : Добавить ребро между вершинами  $u$  и  $v$ . Гарантируется, что такого ребра нет.
2.  $- u v$ : Удалить ребро между  $u$  и  $v$ . Гарантируется, что такое ребро есть.
3.  $?$ : Посчитать количество компонент связности в графе.

Вершины пронумерованы целыми числами от 1 до  $N$ . Во всех запросах  $u \neq v$ .

### Формат выходных данных

Для каждого запроса типа ‘?’, Выведите количество компонент связности в момент запроса.

### Пример

стандартный ввод	стандартный вывод
5 11	5
?	1
+ 1 2	1
+ 2 3	2
+ 3 4	
+ 4 5	
+ 5 1	
?	
- 2 3	
?	
- 4 5	
?	